Sistem *Password* pada *SECURITY ENHANCED LINUX (SELINUX)*

Nurhayati Masthurah
Pusat Penelitian Informatika LIPI
masthurah@informatika.lipi.go.id

Abstrak

Telah dilakukan analisis sistem password SELinux. Untuk melihat keunggulan yang dimiliki oleh sistem password SELinux dibandingkan pada sistem password Linux standard. Percobaan dilakukan di Shell yaitu pada saat sistem operasi login pertama kali, dengan meminta role dan type dari masing-masing user yang akan login. Dari perbandingan kedua sistem diperoleh bahwa SELinux memiliki sistem password dengan konfigurasi menggunakan role dan type dengan permission allow untuk membatasi penggunaan user pada saat login. Konfigurasi ini hanya bisa dirubah oleh administrator.

Kata kunci: SELinux, Sistem Password, Konfigurasi.

1. Pendahuluan

Di era global ini, sistem komputer yang terpasang makin mudah diakses. Menimbulkan banyaknya kelemahan dalam melindungi keamanan sistem untuk menjamin sistem tidak diinterupsi dan diganggu. Sistem Operasi Linux adalah sistem operasi *open source*, dimana *source* yang dimiliki masih terus dikembangkan oleh seluruh pengguna Linux di dunia. Oleh karena itu, *source* selalu diupdate dengan menambahkan *patch* pada sistem. Namun demikian, *patch* yang ada belum mampu menjaga keamanan sistem operasi secara keseluruhan.

Melihat kondisi tersebut, mendorong para peneliti dari NSA (National Security Agency) untuk melakukan penelitian arsitektur sistem operasi yang dapat memberikan fungsi keamanan yang diperlukan oleh sistem secara keseluruhan. *National Security Agency* (NSA) Badan Keamanan milik pemerintah AS, didukung oleh peneliti dari Universitas Utah bekerjasama dengan *Secure Computing Corporation* (SCC), *Network Associates Inc*. (NAI) Labs, dan MITRE *Corporation* pada tanggal 2 Januari 2001 melalui berita di *NSA Press Release* mengeluarkan suatu sistem keamanan yang dijalankan dengan sistem operasi Linux yang diberi nama **SELinux** (*Security Enhanced Linux*)[1]. Solusi keamanan ini mengeluarkan *prototipe* melalui pengembangan konfigurasi *security policy* akses kontrol menggunakan MAC dengan konsep Arsitektur Flask.

SELinux dikondisikan sebagai *source* umum, bentuknya berupa dokumentasi dan *source code* diantara sistem dan beberapa sistem utilitas yang telah dimodifikasi untuk digunakan dari fitur baru. SELinux dilengkapi dengan modul untuk mendukung sistem keamanan yang disebut *Linux Security Modul* (LSM)[2]. SELinux mampu melindungi kernel dari serangan *malicious code.* Sistem SELinux melindungi konfigurasi file dari kerusakan dan sulit merubah konfigurasi *access control* pada sistem.

Dalam makalah ini, penulis akan membahas mengenai sistem *password* yang ada di dalam sistem SELinux, dengan melihat kelebihan yang dimiliki pada sistem password yang ada dibanding sistem Linux standar. Penelitian ini sangat bermanfaat bagi perusahaan, dengan mengatur kebijakan perusahaan dalam setting konfigurasi *security policy* yang dibangun dalam SELinux, sehingga batasan-batasan pengaksesan sistem bisa dikontrol.

2. Perbedaan sistem linux standar dan SELINUX

Security context, sebagai tipe data *policy* merupakan panjang *variabel string* yang diterjemahkan oleh setiap aplikasi atau *user* dengan memahami *security policy*. *Security context* memiliki beberapa atribut yaitu identitas user (*user identity*), TE (*Type Enforcement*) dan *Role Based Access Control* (RBAC). Setiap *subject* dan *object* dalam sistem dikenali sebagai kumpulan atribut keamanan yang disebut *security context*. *Security context* berupa semua atribut keamanan yang diasosiasikan sebagai bagian dari *subject* atau *object* yang relevan dengan *security policy*. Isi dan format dari *security context* tergantung pada bagian *security model* sehingga *security context* hanya diterjemahkan oleh *security server*.

Perbandingan difokuskan pada atribut *access control* Linux standard dengan SELinux. Pada Linux standard, atribut *subject access control* merupakan *user* dan *group* yang sesungguhnya diasosiasikan dengan semua proses berdasarkan struktur proses kernel. Atribut ini dilindungi kernel dan di set berdasarkan jumlah kontrol, termasuk proses login dan program setuid.

Pada SELinux, atribut *access control* memiliki tiga *security context*. Semua *object* dan *subject* dihubungkan dengan *security context*. Dimana, Linux standar menggunakan proses UID (*User Identifier*) atau GID (*Group Identifier*) *file access mode,* dan file UID/GID untuk mengabaikan akses. Fitur *primary access control* SELinux adalah *type enforcement* (TE). *Type* pada TE digunakan untuk menentukan akses. SELinux menambahkan TE pada Linux standar. Hal ini berarti, *access control* Linux standar dan SELinux mampu mengakses *object*. Sebagai contoh, jika SELinux menulis akses ke file tetapi tidak memiliki *permission* w pada file, maka file tidak dapat ditulisi. Perbedaan antara sistem Linux yang sudah di *patch* SELinux dengan sistem Linux standard, terlihat pada tabel 1.

3. Security context

Identifikasi *user* dan *role* diidentifikasikan dalam *security context* yang berdampak dalam kebijakan *access control* pada *type enforcement*. Pada proses, identifikasi *user* dan *role* lebih menarik karena digunakan untuk mengontrol hubungan *type* dengan uid, bahkan dengan *account user* Linux. Pada *object*, identifikasi *user* dan *role* jarang digunakan. *Role* dari *object* disebut object_r, dan *user object* merupakan *user identifier* dari proses yang membuat *object*. Hal ini tidak berdampak pada *access control*.

Tabel 1Perbedaan akses kontrol Linux standard dengan SELinux[4]

Akses Kontrol	**Linux Standar**	**SELinux**
Atribut proses keamanan	Menggunakan UID dan GID asli tanpa perubahan	Menggunakan security context
Atribut objek keamanan	UID dan GID menggunakan mode akses dan mode file	Menggunakan security context
Dasar akses kontrol	roses UID/GID dan mode akses file berdasarkan pada file UID dan GID.	Pengaksesan file dibolehkan menggunakan tipe proses dan tipe file

Perbedaan dilihat pada *user ID* dalam keamanan Linux standar dan *user identifier* dalam *security context SELinux*. Secara tekhnis, hal ini memiliki identifikasi yang berkaitan menggunakan sebagian dari *Linux standard* dan mekanisme *access control* dari peningkatan keamanan. Setiap hubungan ini ditinjau pada proses Login yang dimiliki oleh Linux standar dan SELinux.

3.1 Security context type enforcement

Semua *access control* sistem operasi berdasarkan pada tipe atribut *access control* yang dihubungkan dengan *object* dan *subject*. Dalam SELinux, atribut *access*

control disebut security context. Semua *Object* (*file, interprocess communication channel, socket, network host*, dan lain-lain) dan *Subject* (proses) memiliki *security context* tunggal yang dihubungkan satu sama lain. *Security context* memiliki tiga elemen, yaitu : *user, role* dan *type identifier*. Format ini menspesifikasikan atau menampilkan *security context* sebagai berikut :

```
user : role : type
```

String identifier untuk setiap elemen terletak pada SELinux Policy Language. *Security context* yang valid memiliki *user, role,* dan *type identifier valid*. *Identifier* tersebut didefinisikan oleh penulis *policy*.

3.2 Type enforcement access control

Sistem yang dibangun dengan kernel SELinux tidak mengalami permasalahan dengan batasan user. Pengaksesan kontrol *account user* dapat diarahkan ke pengaksesan kontrol seluruh sistem. Sistem kernel SELinux membatasi user ke dalam domain dengan *access right* dan *permission* yang telah ditentukan. *Security policy* SELinux memiliki konfigurasi yang bervariasi, dengan mengijinkan sistem administrator untuk membuat domain dengan kemampuan yang lebih spesifik. Konfigurasi SELinux dapat dirubah, namun *user* tidak dapat mengakses dari luar untuk dapat mengontrol *domain*.

Dalam SELinux, semua akses secara eksplisit diakui. SELinux memungkinkan tidak ada akses *default*, termasuk UID/GID Linux. Hal ini berarti tidak ada *default superuser* dalam SELinux, berbeda dengan *root* pada Linux standar. Pengaksesan yang diterima dilakukan dengan menspesifikasikan akses dari tipe *subject* (*domain*) dan tipe *object* menggunakan aturan *allow***.** Aturan *allow* memiliki empat elemen, yaitu *Source Type*, berupa *domain type* proses pengaksesan. *Target type*, yaitu tipe dari *object* yang diakses oleh proses. *Object Class*, yaitu *class* dari *object* menentukan akses yang diijinkan. *Permission*, yaitu bentuk akses dari bagian *source type* yang mengijinkan tipe target mengindikasikan *object class*.

Sebagai contoh aturannya diberikan sebagai berikut:

```
allow user_t bin_t : file {read
execute getattr};
```

Hal ini menunjukkan sintaks dasar dari aturan *allow* TE. Aturan ini memiliki dua identifier: *source type* (subject atau domain), user_t; dan *target type* (object), bin_t. *Identifier file* adalah nama dari *object class* yang didefinisikan dalam *policy*. *Permission* berupa bagian dari subset permission valid untuk instance dari file object class. Arti dari aturan tersebut, yaitu user dengan domain type user_t dapat membaca, mengeksekusi, atau mendapatkan atribut file object dengan type bin_t.

4. Metode penelitian

Penelitian dilakukan menggunakan sistem operasi Linux RedHat versi kernel 2.4.18 dengan melakukan patch SELinux, lalu kernel di compile. Sistem dijalankan sebagai shell dengan mengamati konfigurasi security context pada saat Linux login pertama kali. Sistem kerja password dianalisis dengan memperhatikan aturan role dan type yang diminta pada saat login. Sistem password diamati dengan memperhatikan aturan “allow” yang ada pada SELinux.

5. Hasil dan pembahasan

Sintaks umum yang diperoleh pada saat login dengan SELinux adalah :

```
user : role : type atau user :
role : domain
user : user_r : user_t
root : sysadm_r : sysadm_t
```

Semua proses sistem dijalankan dengan *role* `system_r`. sistem ini membagi *user* menjadi dua *role* yaitu `user_r` untuk ordinari *user* dan `sysadm_r` untuk sistem administrator. Setiap *user role* berhubungan dengan inisial *login domain*. *Domain* `user_t` untuk *role* `user_r` dan *domain* `sysadm_t` untuk *role* `sysadm_r`.

5.1 Konfigurasi user

Sintaks deklarasi user, secara umum digambarkan sebagai berikut :

```
user username roles role_set
```

Konfigurasi *user* menentukan setiap *user* yang diakui oleh sistem *security policy* dan menetapkan kumpulan otorisasi *role* untuk setiap *user*. Hanya *user* yang dikenali dalam konfigurasi ini yang dapat digunakan dalam *security context*. Atribut identitas *user* dalam *security context* tidak dirubah oleh *default* ketika program dieksekusi.

Deklarasi login *user*, diberikan dalam sintak berikut :

```
user system_u roles system_r
```

`system_u` sebagai username, system_r sebagai role set. Identitas *user* `system_u` menentukan sistem proses dan objek. Tidak terdapat persamaan identitas *user* Linux untuk `system_u`, dan proses *user* tidak dikenali oleh identitas *user* `system_u`. Identitas *user* ini diotorisasi dengan *role* `system_r`.

Deklarasi login *root*, diberikan dalam sintak berikut :

```
user root roles {user_r sysadm_r}
```

`root` sebagai username, yaitu login dengan menggunakan `root. {user_r sysadm_r}` sebagai system yang dapat diatur oleh `root.`

Deklarasi login *root*, diberikan dalam sintak berikut :

```
user Masth roles {user_r}
```

Masth sebagai nama user yang login. {user_r} sebagai role set, yaitu user ini hanya bisa mengakses usernya sendiri.

Konfigurasi *user* diletakkan di file `users`. Konfigurasi ini menentukan identitas sistem *user*, identitas *user* umum, identitas *user root*, dan beberapa contoh *user* lainnya.

Identitas *user* `user_u` adalah identitas *user* umum untuk *user* Linux yang tidak di daftar dalam identitas *user* SELinux. Modifikasi *daemon* akan menggunakan identitas *user* ini dalam *security context*, apabila *user* umum yang *login* tidak terdaftar sebagai *user* SELinux. Jika administrator tidak menginginkan perizinan akses untuk *user* umum yang tidak dikenal, maka administrator dapat menghapus entri berikut dalam file `users` :

```
user user_u roles user_r
```

Identitas *user* diberikan sebagai proses *user*, ketika memulai *login* pada *shell user*. Identitas *user root* di otorisasi untuk *role* `user_r` dan `sysadm_r`. Proses pada *root* UID (*user identifier*) Linux belum tentu dimiliki identitas *root* SELinux, semenjak identitas *root* SELinux berdiri sendiri.

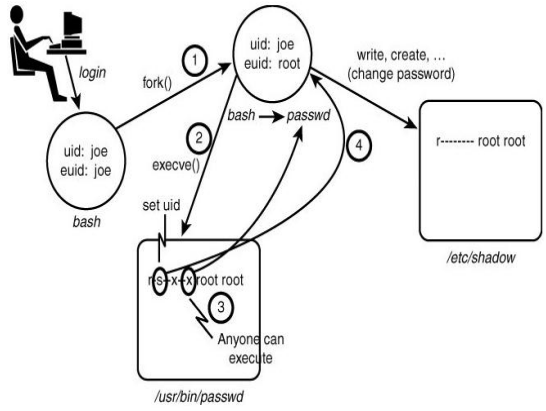
5.2 Sistem kerja password

Sintaks yang menunjukkan konfigurasi sistem password Linux standar, yaitu :

```
# ls -l /usr/bin/passwd
-r-sxx  1 root root 19336 Sep 7
04:11 /usr/bin/passwd
```

Pada sintak diatas menggambarkan s dalam kelompok x ditujukan untuk permission owner. Hal ini juga disebut setuid bit yang berarti setiap proses dapat mengeksekusi file. User root diatas dikenal sebagai file owner, apabila mengeksekusi program password akan dijalankan dengan euid:root.

Arsitektur dari kerja password program security dalam Linux standar diberikan pada gambar 1.

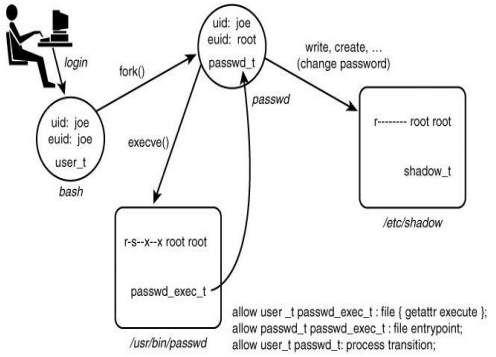


Gambar 1 Arsitektur dari kerja password program security dalam Linux standar (setuid)[4]

Arsitektur kerja *password* pada Gambar 1 menunjukkan *login user* dengan nama Joe, lalu dalam *shell* dijalankan *system call fork()* untuk membuat duplikasi terdekat dari dirinya sendiri. Proses duplikasi uid (*user identifier)* dan euid (*effective user identifier*) Joe, dijalankan pada program *shell (bash)*. Setelah perintah *fork*, proses baru akan menjalankan *system call execve()* untuk

mengeksekusi program *password*. Keamanan Linux standar melakukan pemanggilan uid:joe yang memiliki akses x, karena akses x dihubungkan ke semua orang. Dua kunci penting sebagai hasil kesuksesan *system call execve()*, yaitu pertama, program *shell* dijalankan dalam proses baru yang diletakkan oleh program *password (passwd)*. Kedua, bit *setuid* di set untuk *owner*, maka euid dirubah dari proses original ID ke file *owner* ID (*root*)

Pada sistem SELinux diperlukan adanya transisi domain. Dalam transisi domain diperkenalkan *rule* (aturan) "allow" yang akan digunakan pada tipe proses *domain password* dengan perintah `passwd_t` yang dapat mengakses penyamaran file *password*. Arsitektur kerja password program security dalam SELinux menggunakan transisi domain diberikan pada Gambar 2.



Gambar 2 Password program security in SELinux (domain transitions)

Pada arsitektur ini, *login user* uid dan euid ditambahkan dengan *shell domain* (`user_t`), tipe domain program password (`passwd_t`), dan tipe file shadow password (`shadow_t`). Pada program eksekusi diberikan tipe file eksekusi password (`passwd_exec_t`). Sintaks yang menunjukkan hal tersebut pada sistem, yaitu:

```
# ls -Z /usr/bin/passwd
-r-sxx root root
system_u:object_r:passwd_exec_t
/usr/bin/passwd
```

Pada sintaks diatas menunjukkan bahwa system telah disetting dengan SELinux, terbukti dengan perintah pembacaan direktori `/usr/bin/passwd` yang merupakan direktori penyimpanan setting konfigurasi password user root.

Aturan yang digunakan, seperti pada gambar diatas:

```
allow user_t passwd_exec_t : file
{getattr execute};
allow passwd_t passwd_exec_t :
file entrypoint;
allow user_t passwd_t : process
transition;
```

Dibuat aturan TE policy yang mengijinkan program password dijalankan dengan domain type `passwd_t`. Aturan pertama, yaitu:

```
allow user_t passwd_exec_t : file
{getattr execute};
```

Aturan ini mengijinkan Joe (`user_t`) memulai system call `execve()` pada file eksekusi passwd (`passwd_exec_t`). SELinux mengeksekusi permission file yang memiliki permission yang sama sebagai akses x untuk file dalam Linux standar. Pertama-tama, menjalankan system call `fork()` dengan menduplikasi diri sendiri, termasuk atribut keamanannya. Penggandaan ini menyimpan domain type user Joe (`user_t`). Oleh karena itu, permission yang dieksekusi harus domain original (*domain type shell*).

Aturan kedua:

```
allow passwd_t passwd_exec_t :
file entrypoint;
```

Aturan ini memberikan akses entrypoint ke domain `passwd_t`. Permission entrypoint mendefinisikan file eksekusi yang menjalankan domain. Transisi domain, yaitu domain `passwd_t` memiliki akses entrypoint ke file eksekusi yang digunakan untuk transisi ke domain type baru. Diasumsikan bahwa, eksekusi passwd di beri tanda `passwd_exec_t`. Type `passwd_t` memiliki permission entrypoint ke `passwd_exec_t`. Situasi ini menjelaskan bahwa hanya program password yang dapat dijalankan pada domain type `passwd_t`. Sehingga program ini memiliki *security control* yang baik.

Aturan ketiga:

```
allow user_t passwd_t : process
transition;
```

Aturan `allow` diatas tidak memiliki akses ke object. Object class adalah proses.

Seluruh resource system berada di object class. Aturan ketiga, permission yang dilakukan adalah akses transition. Permission dilakukan untuk mengijinkan tipe security context process di rubah. Tipe umum user (`user_t`) memiliki permission transition ke tipe baru (`passwd_t`) untuk transisi domain yang dijinkan.

Ketiga aturan allow ini menyediakan akses penting untuk transisi domain yang terjadi. Untuk menyukseskan transisi domain, ketiga aturan tersebut harus disertakan. Oleh karena itu, transisi domain diijinkan ketika ketiga aturan ini dijalankan. Proses pertama, domain type baru memiliki akses entrypoint untuk mengeksekusi tipe file. Kedua, domain type baru atau lama memiliki akses eksekusi ke tipe file entrypoint. Ketiga, domain type memiliki akses transition ke domain type baru.

Ketika ketiga permission ini diijinkan dalam policy TE, transisi domain dapat dijalankan. Dengan penggunaan permission entrypoint pada file eksekusi, dimiliki batasan kontrol program yang akan dijalankan dengan memberikan domain type. System call `execve()` adalah satu-satunya cara untuk merubah domain type.

Digunakan shadow password file dan password program. Dengan melakukan pengujian security context dari dua file ini, tipenya berupa `shadow_t` dan `passwd_exec_t`.

`passwd_exec_t` adalah tipe masukan untuk domain `passwd_t`. Perintah ini dijalankan pada dua window, sebagai berikut:

Jendela pertama menjalankan perintah `passwd`:

```
$ passwd
Changing password for user joe.
Changing password for joe
(current) UNIX password:
```

Jendela kedua, menset perintah `su` sebagai `root` dan perintah `ps` :

```
$ su
Password:
Your default context is
root:sysadm_r:sysadm_t.

Do you want to choose a different
one? [n]
# ps axZ|grep passwd
user_u:user_r:passwd_t
4299 pts/1    S+     0:00 passwd
```

6. Kesimpulan

Pada sistem *password* SELinux terdapat konfigurasi *permission* dengan meminta *role* dan *type* yang dimiliki oleh user yang login. Hal ini dilakukan untuk membatasi *login user* bahkan *root* dengan melakukan konfirmasi terlebih dahulu terhadap *role* yang dimiliki oleh masing-masing user. Konfigurasi ini hanya bisa dirubah oleh *login administrator*.

7. Ucapan terima kasih

Penulis mengucapkan terimakasih kepada Bapak Prof. Dr. Masno Ginting atas bimbingan dan arahannya dalam penulisan paper ini.

8. Daftar pustaka

[1] NSA Home Page., 2000, “http://www.nsa.gov/selinux”, Diakses tanggal 26 Juni 2007.

[2] Loscocco, P, A., Smalley, S, D., “*Integrating Flexible Support for Security Policies into The Linux Operating System”*, Technical Report NSA and NAI Labs, 2000.

[3] Masthurah, N., “*Analisis Konfigurasi Security Policy Sistem Operasi dengan Arsitektur Flask pada SELinux”*, Ditulis di Skripsi S1, Program Studi Ilmu Komputer, Universitas Gadjah Mada, 2002.

[4] Mayer, F., MacMillan, K., Caplan, D., “SELinux by Example : Using Security Enhanced Linux”, Prentice Hall, 27 Juli 2006, Pages 456, ISBN 10: 0-131-96369-4, ISBN 13: 978-0-13-196369-